AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions and listing of claims in the above-referenced application.

LISTING OF CLAIMS:

1. (Currently amended) A computer implemented method for automatically tracking build information comprising:

extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said one or more builds;

registering said one or more builds by storing said build information corresponding to each of said one or more builds in a database;

automatically determining software module information about software being tested, wherein said software module information is gathered from one or more software modules during execution of said software being tested; [[and]]

performing a query of the database to retrieve first build information for a first build of the one or more builds in the database; and

~~automatically~~ determining a volatility metric of code change that has occurred between software modules in both said first build information and ~~a first of said one or more builds included in the database which corresponds to~~ said software module information of the software being tested, wherein said determining the volatility metric includes matching portions of the first build information to corresponding portions of said software module information and determining at least one of: a number of functions added, a number of functions removed and a number of functions modified in a software module of the software module information in comparison to a software module of the first build information.

2. (Cancelled)

3. (Previously presented) The method of Claim 1, wherein said database that includes said build information is an object database, and the method further comprises:

creating and storing one or more objects corresponding to each of said builds; and

creating and storing one or more objects corresponding to software modules included in each of the builds.

4. (Previously presented) The method of Claim 3, further including:

creating and storing a session object corresponding to a test session of said software being tested;

creating and storing one or more objects corresponding to software modules describing said software module information;

automatically determining a previously created build object corresponding to one of said one or more builds previously registered; and

storing an address of said previously created build object in said session object.

5. (Cancelled)

6. (Previously presented) The method of Claim 1, wherein said automatically determining software module information further includes:

gathering runtime information about software modules dynamically loaded in a computer system upon which said software being tested is executing.

7. (Original) The method of Claim 6, wherein subroutine calls associated with an operating system executing in said computer system are used in said gathering runtime information.

8. (Previously presented) The method of Claim 1, further comprising:

using said build information in said database to automatically determine a second of said one or more builds corresponding to software module information included in a bug report; and

creating and storing a bug report object corresponding to said bug report, said bug report object including an address associated with said second build.

9. (Original) The method of Claim 8, further including:

submitting said bug report included in a formatted electronic message;

interpreting data included in said formatted electronic message to enable said determining of said second build.

10. (Original) The method of Claim 9, further including:

creating and storing another build object corresponding to a build in which said bug report is identified as being corrected; and

associating said other build object with said bug report object in said database.

11. (Previously presented) The method of Claim 1, wherein said automatically determining said first build further comprises:

determining said first build for which a matching build is being determined from said one or more builds registered;

determining a candidate list including one or more builds having at least one software module in common with said first build;

for each build included in said candidate list, determining if software modules included in said each build match software modules included in said first build;

for each build included in said candidate list, if there are no software modules included in said each build that have a module name and associated attributes matching a software module included in said first build, determining that said each build is not a match for said first build; and

for each build included in said candidate list, if a first of said software modules included in said each build has a module name that matches a software module included in said first build but attributes associated with said software module do not match said software module included in said first build, determining that said each build is not a match for said first build.

12. (Previously presented) The method of Claim 11, further including:

for each build included in said candidate list, determining a number of matches for software modules included in said each build having a matching module name and attributes of a software module included in said first build;

determining a valid list of one or more matching builds, each of said builds included in said valid list having a full match for software modules included in said each build with software modules included in said first build, each software module included in said each build having a corresponding matching software module included in said first build, said name and associated attributes of said each software module matching said matching software module included in said first build; and

determining a maybe list of one or more builds, each of said one or more builds included in said maybe list having at least one software module having a module name that does not have a matching software module included in said first build, said each build including at least one software module having a module name and associated attributes matching another software module included in said first build, wherein said each build has an associated number of matches.

13. (Original) The method of Claim 12, further including:

if said valid list is empty, for each project, adding a build from said maybe list to said valid list in which the build added has a maximum number of matches of builds associated with said each project, a project having one or more associated builds;

performing an alternative action if said valid list is empty;

if there is only one build included in said valid list, determining that said one build matches said first build; and

if there is more than one build included in said valid list, selecting one of the more than one builds included in said valid list as being the build matching said first build.

14. (Original) The method of Claim 13, further including:

selecting one or more build associated with a software project.

15. (Original) The method of Claim 13, further including:

determining a matching build in accordance with a predetermined build selected from a list of more than one build previously registered.

16. (Currently amended) A computer implemented method for determining a code volatility metric, the method comprising:

extracting build information by processing, for at least two builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said at least two builds;

registering said at least two builds by storing said build information corresponding to each of said at least two builds in a database;

identifying, by retrieving at least a portion of said build information from the database, a first and a second of said at least two builds;

performing a query of the database to determine code volatility between software modules included in both the first and the second of said at least two builds; and

calculating, in response to said query, said code volatility metric using said build information including software module information about said first and said second builds included in the database, said code volatility metric being determined using one or more metrics representing an amount of code change that has occurred between software modules in both said first build and said second build, the using one or more metrics including determining at least one of: a number of functions added, a number of functions removed and a number of functions modified in a software module of the second build in comparison to a software module of the first build.

17. (Previously presented) The method of Claim 16, further comprising:

determining a total number of functions of said first build and said second build;

determining a percentage of functions added in accordance with said total number of functions;

determining a percentage of functions removed in accordance with said total number of functions;

determining a percentage of functions modified in accordance with said total number of functions; and

determining said code volatility metric as a sum of said percentage of functions added, said percentage of functions removed, and said percentage of functions modified.

18. (Cancelled)

19. (Currently Amended) The method of Claim [[19]] <u>16</u>, further including:

determining function checksum information used in determining differences in a function in which a first version of a function is associated with one software module and a second version of the function is associated with a second software module, said first software module being associated with a first build, and said second software module being associated with a second build.

20. (Currently amended) A computer implemented method for tracking build information comprising:

extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said one or more builds;

registering said one or more builds by storing said build information in a database;

executing a software program that includes one or more software modules, said executing including processing said one or more software modules of the software program during execution of said software program to determine module information about said one or more software modules of said software program;

automatically determining, using said build information included in the database and said module information obtained from said executing, a matching build for said one or more software modules of the software program that are also associated with one of said builds previously registered; and

determining testing information associated with the matching build by performing a query of said database, said testing information including at least one type of runtime analysis performed for the matching build, the at least one type of runtime analysis including determining a volatility metric of code change that has occurred between software modules in both said matching build and said module information obtained from said executing, wherein said determining the volatility metric includes matching portions of the matching build to corresponding portions of the module information obtained from said executing and determining at least one of: a number of functions added, a number of functions removed and a number of functions modified in a software module of the module information obtained from said executing in comparison to a software module of the matching build.

21. (Cancelled)

22. (Currently amended) A computer readable medium comprising machine executable code stored thereon for automatically tracking build information, the computer readable medium comprising:

machine executable code for extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said one or more builds;

machine executable code for registering said one or more builds by storing said build information corresponding to each of said one or more builds in a database;

machine executable code for automatically determining software module information about software being tested, wherein said software module information is gathered from one or more software modules during execution of said software being tested; and

machine executable code for determining a first of said one or more builds included in the database which corresponds to said software module information about the software being tested, the determining including determining a volatility metric of code change that has occurred between software modules in both said one or more builds included in the database and said software module information about the software being tested, wherein said determining the volatility metric includes matching portions of the one or more builds included in the database to corresponding portions of the software module information about the software being tested and determining at least one of: a number of functions added, a number of functions removed and a number of functions modified in a software module of the software module information about the software being tested in comparison to a software module of the one or more builds included in the database.

23. (Cancelled)

24. (Previously presented) The computer readable medium of Claim 22, wherein said database that includes said build information is an object database, and the computer program product further comprises:

machine executable code for creating and storing one or more objects corresponding to each of said builds; and

machine executable code for creating and storing one or more objects corresponding to software modules included in each of the builds.

25. (Previously presented) The computer readable medium of Claim 24, further including:

machine executable code for creating and storing a session object corresponding to a test session of said software being tested ;

machine executable code for creating and storing one or more objects corresponding to software modules describing said software module information;

machine executable code for determining automatically a previously created build object corresponding to one of said one or more builds previously registered; and machine executable code for storing an address of said previously created build object in said session object.

26. (Cancelled)

27. (Previously presented) The computer readable medium of Claim 22, wherein said machine executable code for automatically determining software module information further includes:

machine executable code for gathering runtime information about software modules dynamically loaded in a computer system upon which said software being tested is executing.

28. (Previously presented) The computer readable medium of Claim 27, wherein machine executable code associated with an operating system is used in gathering runtime information.

29. (Previously presented) The computer readable medium of Claim 22, further including:

machine executable code for using said build information in said database to automatically determine a second of said one or more builds corresponding to software module information included in a bug report; and

machine executable code for creating and storing a bug report object corresponding to said bug report, said bug report object including an address associated with said second build.

30. (Previously presented) The computer readable medium of Claim 29, further including:

machine executable code for submitting a bug report included in a formatted electronic message; and

machine executable code for interpreting data included in said formatted electronic message to enable determination of said second build.

31. (Previously presented) The computer readable medium of Claim 30, further including:

machine executable code for creating and storing another build object corresponding to a build in which said bug report is identified as being corrected; and

machine executable code for associating said other build object with said bug report object in said database.

32. (Previously presented) The computer readable medium of Claim 22, wherein said machine executable code for automatically determining said first build further comprises:

machine executable code for determining said first build for which a matching build is being determined from said one or more builds registered;

machine executable code for determining a candidate list including one or more builds having at least one software module in common with said first build;

machine executable code for determining, for each build included in said candidate list, if software modules included in said each build match software modules included in said first build;

machine executable code for determining, for each build included in said candidate list, if there are no software modules included in said each build that have a module name and associated attributes matching a software module included in said first build, determining that said each build is not a match for said first build; and

machine executable code for determining, for each build included in said candidate list, that said each build is not a match for said first build if a first of said software modules included in said each build has a module name that matches a software module included in said first build but attributes associated with said software module do not match said software module included in said first build.

33. (Previously presented) The computer readable medium of Claim 22, further including:

machine executable code for determining, for each build included in said candidate list, a number of matches for software modules included in said each build having a matching module name and attributes of a software module included in said first build;

machine executable code for determining a valid list of one or more matching builds, each of said builds included in said valid list having a full match for software modules included in said each build with software modules included in said first build, each software module included in said each build having a corresponding matching software module included in said first build, said name and associated attributes of said each software module matching said matching software module included in said first build; and

machine executable code for determining a maybe list of one or more builds, each of said one or more builds included in said maybe list having at least one software module having a module name that does not have a matching software module included in said first build, said each build including at least one software module having a module name and associated attributes matching another software module included in said first build, wherein said each build has an associated number of matches.

34. (Previously presented) The computer readable medium of Claim 33, further including:

machine executable code for adding, if said valid list is empty, for each project, a build from said maybe list to said valid list in which the build added has a maximum number of matches of builds associated with said each project, a project having one or more associated builds;

machine executable code for performing an alternative action if said valid list is empty;

machine executable code for determining that said one build matches said first build if there is only one build included in said valid list; and

machine executable code for selecting one of the more than one builds included in said valid list as being the build matching said first build if there is more than one build included in said valid list.

35. (Previously presented) The computer readable medium of Claim 34, further including:

machine executable code for selecting one or more build associated with a software project.

36. (Previously presented) The computer readable medium of Claim 34, further including:

machine executable code for determining a matching build in accordance with a predetermined build selected from a list of more than one build previously registered.

37. (Currently amended) A computer readable medium comprising machine executable code stored thereon for determining a code volatility metric, the computer readable medium comprising:

machine executable code for extracting build information by processing, for at least two builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said at least two builds;

machine executable code for registering said at least two builds by storing said build information corresponding to each of said at least two builds in a database;

machine executable code for identifying, by retrieving at least a portion of said build information from the database, a first and a second of said at least two builds;

machine executable code for performing a query of the database to determine code volatility between software modules included in both the first and the second of said at least two builds; and

machine executable code for calculating, in response to said query, said code volatility metric using said build information including software module information about said first and said second builds included in the database, said code volatility metric being determined using one or more metrics representing an amount of code change that has occurred between software modules in both said first build and said second build, the using one or more metrics including determining at least one of: a number of functions added, a number of functions removed and a

<u>number of functions modified in at least one software module of the second build in a comparison to at least one software module of the first build</u>.

38. (Previously presented) The computer readable medium of Claim 37, further comprising:

machine executable code for determining a total number of functions of said first build and said second build;

machine executable code for determining a percentage of functions added in accordance with said total number of functions;

machine executable code for determining a percentage of functions removed in accordance with said total number of functions;

machine executable code for determining a percentage of functions modified in accordance with said total number of functions; and

machine executable code for determining said code volatility metric as a sum of said percentage of functions added, said percentage of functions removed, and said percentage of functions modified.

39. (Cancelled)

40. (Previously presented) The computer readable medium of Claim 38, further including:

machine executable code for determining function signature information used in determining differences in a function in which a first version of a function is associated with one software module and a second version of the function is associated with a second software module, said first software module being associated with a first build, and said second software module being associated with a second build.

41. (Currently amended) A computer readable medium comprising executable code stored thereon for tracking build information, the computer readable medium comprising:

machine executable code for extracting build information by processing, for each of one or more builds, one or more software modules produced using a compilation process resulting in said one or more software modules, wherein said build information extracted includes software module information about at least one software module produced as an output of a compilation process for each of said one or more builds;

machine executable code for registering said one or more builds by storing said build information in a database;

machine executable code for executing a software program that includes one or more software modules, said executing including processing said one or more software modules of the software program during execution of said software program to determine module information about said one or more software modules of said software program;

machine executable code for automatically determining, using build information included in the database and said module information obtained from said executing, a matching build for said one or more software modules of the software program that are also associated with one of said builds previously registered; and

machine executable code for determining testing information associated with the matching build by performing a query of said database, said testing information including at least one type of runtime analysis performed for the matching build, the at least one type of runtime analysis including determining a volatility metric of code change that has occurred between software modules in both said matching build and said module information obtained from said executing, wherein said determining the volatility metric includes matching portions of the matching build to corresponding portions of the module information obtained from said executing and determining at least one of: a number of functions added, a number of functions removed and a number of functions modified in a software module of the module information obtained from said executing in comparison to a software module of the matching build.

42. (Cancelled)